

Network Working Group
INTERNET-DRAFT
Obsoletes: RFC [1778](#)

M. Wahl
Critical Angle Inc.
A. Coulbeck
ISODE Consortium
T. Howes
Netscape Communications Corp.
S. Kille
ISODE Consortium
October 22, 1996

Intended Category: Standards Track

**Lightweight Directory Access Protocol:
Standard and Pilot Attribute Definitions
<[draft-ietf-asid-ldapv3-attributes-03.txt](#)>**

[1.](#) Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "lidl-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

[2.](#) Abstract

The Lightweight Directory Access Protocol (LDAP) [[1](#)] requires that the contents of AttributeValue fields in protocol elements be octet strings. This document defines the requirements that must be satisfied by encoding rules used to render directory attribute syntaxes into a form suitable for use in the LDAP, then goes on to define the encoding rules for the standard set of attribute syntaxes of [[2](#)], [[3](#)] and [[4](#)]. It also identifies all the attribute types, object classes and matching rules for LDAP version 3.

[3.](#) Overview

[Section 4](#) states the general requirements and notations for attribute types, object classes, syntax and matching rule definitions.

The core definitions are given in [section 5](#), those which are based on X.500(1993) in [section 6](#), and other optional definitions in [section 7](#).

4. General Issues

4.1. Attribute Types

The attribute types are described by sample values for the subschema "attributeTypes" attribute, which is written in the AttributeTypeDescription syntax. While lines have been folded for readability, the values transferred in protocol would not contain newlines.

The AttributeTypeDescription is encoded according to the following BNF, and the productions for <oid>, <DirectoryStrings> and <DirectoryString> are given in sections [4.2.1](#).

```
<AttributeTypeDescription> ::= "("
    <oid>      -- AttributeType identifier
    [ "NAME" <DirectoryStrings> ] -- name used in AttributeType
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    [ "SUP" <oid> ]           -- derived from this other AttributeType
    [ "EQUALITY" <oid> ]      -- Matching Rule name
    [ "ORDERING" <oid> ]      -- Matching Rule name
    [ "SUBSTR" <oid> ]         -- Matching Rule name
    [ "SYNTAX" <DirectoryString> ] -- see section 4.2
    [ "SINGLE-VALUE" ]         -- default multi-valued
    [ "COLLECTIVE" ]           -- default not collective
    [ "NO-USER-MODIFICATION" ] -- default user modifiable
    [ "USAGE" <AttributeUsage> ] -- default user applications
    ")"

<AttributeUsage> ::=
    "userApplications"
|   "directoryOperation"
|   "distributedOperation" -- DSA-shared
|   "dSAOperation"         -- DSA-specific, value depends on server
```

Servers are not required to provide the same or any text in the description part of the subschema values they maintain.

Servers must implement all the attribute types in [section 5.1](#), and may also implement the types listed in sections [6.1](#) and [7.1](#). Servers must be able to perform equality matching of values, but need not perform any additional validity checks on attribute values.

Servers may recognize additional names and attributes not listed in this document. Later documents may define additional types.

Servers may implement additional attribute types not listed in this document, and if they do so, must publish the definitions of the types in the attributeTypes attribute of their subschema subentries.

AttributeDescriptions may be used as the value in a NAME part of an AttributeTypeDescription. Note that these are case insensitive.

4.2. Syntaxes

This section defines general requirements for LDAP attribute value syntax encodings. All documents defining attribute syntax encodings for use with LDAP are expected to conform to these requirements.

The encoding rules defined for a given attribute syntax must produce octet strings. To the greatest extent possible, encoded octet strings should be usable in their native encoded form for display purposes. In particular, encoding rules for attribute syntaxes defining non-binary values should produce strings that can be displayed with little or no translation by clients implementing LDAP. There are a few cases (e.g. Audio) however, when it is not sensible to produce a printable representation, and clients must not assume that an unrecognized syntax is a string representation.

4.2.1. Common Encoding Aspects

In these encodings where an arbitrary string is used as part of a larger production (other than a Distinguished Name), a backslash quoting mechanism is used to encode the following separator symbol character (such as "'", '\$' or '#') if it should occur in that string. The backslash is followed by a pair of hexadecimal digits representing the next character. A backslash itself in the string which forms part of a larger syntax is always transmitted as '\5C' or '\5c'.

For the purposes of defining the encoding rules for attribute syntaxes, the following auxiliary BNF definitions will be used:

```

<a> ::= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' |
        'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' |
        's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | 'A' |
        'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' |
        'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' |
        'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'

<d> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

<hex-digit> ::= <d> | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' |
                'A' | 'B' | 'C' | 'D' | 'E' | 'F'

<k> ::= <a> | <d> | '-'

<p> ::= <a> | <d> | '"' | '(' | ')' | '+' | ',' | '-' | '.' |
        '/' | ':' | '?' | ' '

<letterstring> ::= <a> | <a> <letterstring>

<numericstring> ::= <d> | <d> <numericstring>

```

$\langle \text{keystring} \rangle ::= \langle a \rangle \mid \langle a \rangle \langle \text{anhstring} \rangle$

$\langle \text{anhstring} \rangle ::= \langle k \rangle \mid \langle k \rangle \langle \text{anhstring} \rangle$

```
<printablestring> ::= <p> | <p> <printablestring>

<space> ::= ' ' | ' ' <space>

<whsp> ::= <space> | empty

<utf8> ::= any sequence of octets formed from the UTF-8 [11]
           transformation of a character from ISO 10646 [12]

<dstring> ::= <utf8> | <utf8> <dstring>

<DirectoryStrings> ::= <DirectoryString> | '(' <DirectoryStringList> ')'

<DirectoryStringList> ::= <DirectoryStringList> <DirectoryString> | ""

<DirectoryString> ::= ''' <dstring> '''

<oids> ::= <oid> | '(' <oidlist> ')'

<oidlist> ::= <oidlist> '$' <oid> | <oid>

-- <oid> is defined in 5.2.1.15
```

4.2.2 Binary Transfer of Values

This encoding format is used if the binary encoding is requested by the client for an attribute, or if the attribute syntax name is 'Binary'. The value, an instance of the ASN.1 AttributeValue type, is BER-encoded, subject to the restrictions of section 5.1 of [1], and this sequence of octets is used as the value.

All servers must implement this form for both generating Search responses and parsing Add, Compare and Modify requests. Clients must be prepared receiving values in binary (e.g. userCertificate or audio), and must not simply display binary or unrecognized values to users.

4.2.3 Syntax Names

Names of syntaxes for use with LDAP are ASCII strings which either begin with a letter and contain only letters or digits. The names are case insensitive. Historically since syntaxes correspond to ASN.1 types, they have been named starting with a capital letter. A suggested upper bound on the number of characters in value with a DirectoryString or IA5String syntax or the number of bytes in a value for all other syntaxes may be indicated by appending this bound count inside of curly braces, e.g. "DirectoryString{64}". Note that a single character of the DirectoryString may be encoded in more than one byte since UTF-8 is a variable-length encoding.

Syntax names do not have global scope: two clients or servers may

know of different syntaxes with the same name.

The definition of additional arbitrary syntaxes is strongly depreciated since it will hinder interoperability: today's client and server implementations generally do not have the ability to dynamically recognize new syntaxes. In most cases attributes will be defined with the DirectoryString syntax.

The following syntax names are used for attributes in this document. Servers are only required to implement the syntaxes in [section 5.2](#).

AccessPoint	ACIItem
AttributeTypeDescription	Audio
Binary	BitString
Certificate	CertificateList
CertificatePair	DataQualitySyntax
DeliveryMethod	DirectoryString
DITContentRuleDescription	DN
DSAQualitySyntax	DSEType
EnhancedGuide	FacsimileTelephoneNumber
Fax	GeneralizedTime
Guide	IA5String
INTEGER	JPEG
MailPreference	MasterAndShadowAccessPoints
MatchingRuleDescription	MatchingRuleUseDescription
ModifyRight	NameAndOptionalUID
NameFormDescription	NumericString
ObjectClassDescription	OID
OtherMailbox	Password
PostalAddress	PresentationAddress
PrintableString	ProtocolInformation
SubtreeSpecification	SupplierAndConsumers
SupplierInformation	SupplierOrConsumer
TelephoneNumber	TeletexTerminalIdentifier
TelexNumber	UTCTime

[4.3. Object Classes](#)

These are described as sample values for the subschema "objectClasses" attribute for a server which implements the LDAP schema. While lines have been folded for readability, the values transferred in protocol would not contain newlines.

Object class descriptions are written according to the following BNF:

```
<ObjectClassDescription> ::= "("
    <oid>    -- ObjectClass identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    [ "SUP" <oids> ]    -- Superior ObjectClasses
```

```
[ ( "ABSTRACT" | "STRUCTURAL" | "AUXILIARY" ) ] -- default structural
[ "MUST" <oids> ]    -- AttributeTypes
[ "MAY" <oids> ]      -- AttributeTypes
")"
```

Servers must implement all the object classes in [section 5.3](#):

account	alias
applicationEntity	applicationProcess
certificationAuthority	country
dNSDomain	dSA
device	document
documentSeries	domain
domainRelatedObject	friendlyCountry
groupOfNames	groupOfUniqueNames
locality	newPilotPerson
organization	organizationalPerson
organizationalRole	organizationalUnit
person	pilotDSA
pilotObject	pilotOrganization
qualityLabelledData	rFC822localPart
residentialPerson	room
simpleSecurityObject	strongAuthenticationUser
top	

and may also implement the object classes of 6.3 and 7.3.

Servers may implement additional object classes not listed in this document, and if they do so, must publish the definitions of the classes in the objectClasses attribute of their subschema subentries. Later documents may define additional object classes.

[4.4. Matching Rules](#)

Matching rules are used by servers to compare attribute values against assertion values when performing Search and Compare operations.

Most of the attributes given in this document will have an equality matching rule defined.

Matching rule descriptions are written according to the following BNF:

```
<MatchingRuleDescription> ::= "("
    <oid>    -- MatchingRule identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    "SYNTAX" <DirectoryString>
    ")"
```

Servers must implement all the matching rules in [section 5.4](#):

bitStringMatch	caseExactIA5Match
caseIgnoreIA5Match	caseIgnoreListMatch
caseIgnoreMatch	distinguishedNameMatch
generalizedTimeMatch	integerMatch

numericStringMatch
octetStringMatch

objectIdentifierMatch
telephoneNumberMatch

and may also implement the matching rules of 6.4 and 7.4.

Servers may implement additional matching rules not listed in this document, and if they do so, must publish the definitions of the matching rules in the matchingRules attribute of their subschema subentries.

5. Mandatory Definitions

[Section 5](#) contains definitions which must be implemented by all servers.

5.1. Attribute Types

Servers must recognize all the attributes of this section (5.1.1 - 5.1.5).

5.1.1. Standard User Attributes

The attributes listed in this section are those defined in X.520(1993), likely to be present in user entries. Servers must recognize all the attributes of this section. The semantics of attributes 2.5.4.0 through 2.5.4.40 are summarized in [RFC 1274](#).

```
( 2.5.4.0 NAME 'objectClass' EQUALITY objectIdentifierMatch SYNTAX 'OID' )
```

```
( 2.5.4.1 NAME 'aliasedObjectName' EQUALITY distinguishedNameMatch  
  SYNTAX 'DN' SINGLE-VALUE )
```

```
( 2.5.4.2 NAME 'knowledgeInformation' EQUALITY caseIgnoreMatch  
  SYNTAX 'DirectoryString{32768}' )
```

```
( 2.5.4.3 NAME 'cn' SUP name )
```

```
( 2.5.4.4 NAME 'sn' SUP name )
```

```
( 2.5.4.5 NAME 'serialNumber' EQUALITY caseIgnoreMatch  
  SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'PrintableString{64}' )
```

```
( 2.5.4.6 NAME 'c' SUP name SINGLE-VALUE )
```

```
( 2.5.4.7 NAME 'l' SUP name )
```

```
( 2.5.4.8 NAME 'st' SUP name )
```

```
( 2.5.4.9 NAME 'street' EQUALITY caseIgnoreMatch  
  SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{128}' )
```

```
( 2.5.4.10 NAME 'o' SUP name )
```

```
( 2.5.4.11 NAME 'ou' SUP name )
```

```
( 2.5.4.12 NAME 'title' SUP name )
```

```
( 2.5.4.13 NAME 'description' EQUALITY caseIgnoreMatch  
  SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{1024}' )
```

- (2.5.4.14 NAME 'searchGuide' SYNTAX 'Guide')
- (2.5.4.15 NAME 'businessCategory' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{128}')
- (2.5.4.16 NAME 'postalAddress' EQUALITY caseIgnoreListMatch
SUBSTRINGS caseIgnoreListSubstringsMatch SYNTAX 'PostalAddress')
- (2.5.4.17 NAME 'postalCode' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{40}')
- (2.5.4.18 NAME 'postOfficeBox' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{40}')
- (2.5.4.19 NAME 'physicalDeliveryOfficeName' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{128}')
- (2.5.4.20 NAME 'telephoneNumber' EQUALITY telephoneNumberMatch
SUBSTRINGS telephoneNumberSubstringsMatch SYNTAX 'TelephoneNumber{32}')
- (2.5.4.21 NAME 'telexNumber' SYNTAX 'TelexNumber')
- (2.5.4.22 NAME 'teletexTerminalIdentifier'
SYNTAX 'TeletexTerminalIdentifier')
- (2.5.4.23 NAME 'facsimileTelephoneNumber'
SYNTAX 'FacsimileTelephoneNumber')
- (2.5.4.24 NAME 'x121Address' EQUALITY numericStringMatch
SUBSTRINGS numericStringSubstringsMatch SYNTAX 'NumericString{15}')
- (2.5.4.25 NAME 'internationalISDNNumber' EQUALITY numericStringMatch
SUBSTRINGS numericStringSubstringsMatch SYNTAX 'NumericString{16}')
- (2.5.4.26 NAME 'registeredAddress' SUP postalAddress
SYNTAX 'PostalAddress')
- (2.5.4.27 NAME 'destinationIndicator' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'PrintableString{128}')
- (2.5.4.28 NAME 'preferredDeliveryMethod' SYNTAX 'DeliveryMethod'
SINGLE-VALUE)
- (2.5.4.29 NAME 'presentationAddress' EQUALITY presentationAddressMatch
SYNTAX 'PresentationAddress' SINGLE-VALUE)
- (2.5.4.30 NAME 'supportedApplicationContext'
EQUALITY objectIdentifierMatch SYNTAX 'OID')
- (2.5.4.31 NAME 'member' SUP distinguishedName)

(2.5.4.32 NAME 'owner' SUP distinguishedName)


```
( 2.5.4.33 NAME 'roleOccupant' SUP distinguishedName )

( 2.5.4.34 NAME 'seeAlso' SUP distinguishedName )

( 2.5.4.35 NAME 'userPassword' EQUALITY octetStringMatch
  SYNTAX 'Password{128}' )

( 2.5.4.36 NAME 'userCertificate' SYNTAX 'Certificate' )

( 2.5.4.37 NAME 'cACertificate' SYNTAX 'Certificate' )

( 2.5.4.38 NAME 'authorityRevocationList' SYNTAX 'CertificateList' )

( 2.5.4.39 NAME 'certificateRevocationList' SYNTAX 'CertificateList' )

( 2.5.4.40 NAME 'crossCertificatePair' SYNTAX 'CertificatePair' )

( 2.5.4.41 NAME 'name'
  DESC 'The name attribute type is the attribute supertype from which
        string attribute types typically used for naming may be formed.'
  EQUALITY caseIgnoreMatch
  SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{32768}' )

( 2.5.4.42 NAME 'givenName' SUP name )

( 2.5.4.43 NAME 'initials'
  DESC 'The initials attribute type contains the initials of some or all
        of an individuals names, but not the surname(s).'
  SUP name )

( 2.5.4.44 NAME 'generationQualifier'
  DESC 'e.g. Jr or II.'
  SUP name )

( 2.5.4.45 NAME 'x500UniqueIdentifier'
  DESC 'used to distinguish between objects when a distinguished name has
        been reused.'
  EQUALITY bitStringMatch SYNTAX 'BitString' )

( 2.5.4.46 NAME 'dnQualifier'
  DESC 'The dnQualifier attribute type specifies disambiguating
        information to add to the relative distinguished name of an
        entry. It is intended to be used for entries held in multiple
        DSAs which would otherwise have the same name, and that its
        value be the same in a given DSA for all entries to which this
        information has been added.'
  EQUALITY caseIgnoreMatch
  ORDERING caseIgnoreOrderingMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'PrintableString' )
```

(2.5.4.47 NAME 'enhancedSearchGuide' SYNTAX 'EnhancedGuide')

- (2.5.4.48 NAME 'protocolInformation' EQUALITY protocolInformationMatch
SYNTAX 'ProtocolInformation')
- (2.5.4.49 NAME 'distinguishedName'
DESC 'This is not the name of the object itself, but a base type
from which attributes with DN syntax inherit.'
EQUALITY distinguishedNameMatch
SYNTAX 'DN')
- (2.5.4.50 NAME 'uniqueMember' EQUALITY uniqueMemberMatch
SYNTAX 'NameAndOptionalUID')
- (2.5.4.51 NAME 'houseIdentifier' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{32768}')

5.1.2. Pilot User Attributes

These attributes are defined in [RFC 1274](#). Servers must recognize all the attributes of this section.

- (0.9.2342.19200300.100.1.1 NAME 'uid' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.2 NAME 'textEncoded0Raddress'
EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.3 NAME 'mail' EQUALITY caseIgnoreIA5Match
SUBSTRINGS caseIgnoreIA5SubstringsMatch SYNTAX 'IA5String{256}')
- (0.9.2342.19200300.100.1.4 NAME 'info' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{2048}')
- (0.9.2342.19200300.100.1.5 NAME 'drink' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.6 NAME 'roomNumber' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.7 NAME 'photo' SYNTAX 'Fax{250000}')
- (0.9.2342.19200300.100.1.8 NAME 'userClass' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.9 NAME 'host' EQUALITY caseIgnoreMatch
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}')
- (0.9.2342.19200300.100.1.10 NAME 'manager'
EQUALITY distinguishedNameMatch SYNTAX 'DN')

```
( 0.9.2342.19200300.100.1.11 NAME 'documentIdentifier'  
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch  
  SYNTAX 'DirectoryString{256}' )
```

```
( 0.9.2342.19200300.100.1.12 NAME 'documentTitle' EQUALITY caseIgnoreMatch
  SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString{256}' )

( 0.9.2342.19200300.100.1.13 NAME 'documentVersion'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString{256}' )

( 0.9.2342.19200300.100.1.14 NAME 'documentAuthor'
  EQUALITY distinguishedNameMatch SYNTAX 'DN' )

( 0.9.2342.19200300.100.1.15 NAME 'documentLocation'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString{256}' )

( 0.9.2342.19200300.100.1.20 NAME 'homePhone' EQUALITY telephoneNumberMatch
  SUBSTRINGS telephoneNumberSubstringsMatch SYNTAX 'TelephoneNumber{32}' )

( 0.9.2342.19200300.100.1.21 NAME 'secretary'
  EQUALITY distinguishedNameMatch SYNTAX 'DN' )

( 0.9.2342.19200300.100.1.22 NAME 'otherMailbox' SYNTAX 'OtherMailbox' )

( 0.9.2342.19200300.100.1.25 NAME 'dc' EQUALITY caseIgnoreIA5Match
  SUBSTRINGS caseIgnoreIA5SubstringsMatch SYNTAX 'IA5String' )

( 0.9.2342.19200300.100.1.26 NAME 'dnsRecord'
  EQUALITY caseExactIA5Match SYNTAX 'IA5String' )

( 0.9.2342.19200300.100.1.37 NAME 'associatedDomain'
  EQUALITY caseIgnoreIA5Match SUBSTRINGS caseIgnoreIA5SubstringsMatch
  SYNTAX 'IA5String' )

( 0.9.2342.19200300.100.1.38 NAME 'associatedName'
  EQUALITY distinguishedNameMatch SYNTAX 'DN' )

( 0.9.2342.19200300.100.1.39 NAME 'homePostalAddress'
  EQUALITY caseIgnoreListMatch
  SUBSTRINGS caseIgnoreListSubstringsMatch SYNTAX 'PostalAddress' )

( 0.9.2342.19200300.100.1.40 NAME 'personalTitle'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString' )

( 0.9.2342.19200300.100.1.41 NAME 'mobile' EQUALITY telephoneNumberMatch
  SUBSTRINGS telephoneNumberSubstringsMatch SYNTAX 'TelephoneNumber{32}' )

( 0.9.2342.19200300.100.1.42 NAME 'pager' EQUALITY telephoneNumberMatch
  SUBSTRINGS telephoneNumberSubstringsMatch SYNTAX 'TelephoneNumber{32}' )

( 0.9.2342.19200300.100.1.43 NAME 'co' EQUALITY caseIgnoreMatch
```

```
SUBSTRINGS caseIgnoreSubstringsMatch SYNTAX 'DirectoryString' )
```

```
( 0.9.2342.19200300.100.1.44 NAME 'uniqueIdentifier'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString' )

( 0.9.2342.19200300.100.1.45 NAME 'organizationalStatus'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString{256}' )

( 0.9.2342.19200300.100.1.46 NAME 'janetMailbox'
  EQUALITY caseIgnoreIA5Match SUBSTRINGS caseIgnoreIA5SubstringsMatch
  SYNTAX 'IA5String{256}' )

( 0.9.2342.19200300.100.1.47 NAME 'mailPreferenceOption'
  SYNTAX 'INTEGER' SINGLE-VALUE )

( 0.9.2342.19200300.100.1.48 NAME 'buildingName'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString{256}' )

( 0.9.2342.19200300.100.1.49 NAME 'dsaQuality'
  SYNTAX 'DSAQualitySyntax' SINGLE-VALUE )

( 0.9.2342.19200300.100.1.50 NAME 'singleLevelQuality'
  SYNTAX 'DataQualitySyntax' SINGLE-VALUE )

( 0.9.2342.19200300.100.1.51 NAME 'subtreeMinimumQuality'
  SYNTAX 'DataQualitySyntax' SINGLE-VALUE )

( 0.9.2342.19200300.100.1.52 NAME 'subtreeMaximumQuality'
  SYNTAX 'DataQualitySyntax' SINGLE-VALUE )

( 0.9.2342.19200300.100.1.53 NAME 'personalSignature'
  SYNTAX 'Fax{50000}' )

( 0.9.2342.19200300.100.1.54 NAME 'dITRedirect'
  EQUALITY distinguishedNameMatch SYNTAX 'DN' )

( 0.9.2342.19200300.100.1.55 NAME 'audio' SYNTAX 'Audio{250000}' )

( 0.9.2342.19200300.100.1.56 NAME 'documentPublisher'
  EQUALITY caseIgnoreMatch SUBSTRINGS caseIgnoreSubstringsMatch
  SYNTAX 'DirectoryString' )

( 0.9.2342.19200300.100.1.60 NAME 'jpegPhoto' SYNTAX 'JPEG' )
```


5.1.3. Standard Operational Attributes

All servers must recognize the attribute types defined in this section.

- (2.5.18.1 NAME 'createTimestamp' EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch SYNTAX 'GeneralizedTime'
SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation)
- (2.5.18.2 NAME 'modifyTimestamp' EQUALITY generalizedTimeMatch
ORDERING generalizedTimeOrderingMatch SYNTAX 'GeneralizedTime'
SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation)
- (2.5.18.3 NAME 'creatorsName' EQUALITY distinguishedNameMatch SYNTAX 'DN'
SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation)
- (2.5.18.4 NAME 'modifiersName' EQUALITY distinguishedNameMatch SYNTAX 'DN'
SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation)
- (2.5.18.10 NAME 'subschemaSubentry'
DESC 'The value of this attribute is the name of a subschema subentry,
an entry in which the server makes available attributes specifying
the schema.'
EQUALITY distinguishedNameMatch SYNTAX 'DN' NO-USER-MODIFICATION
SINGLE-VALUE USAGE directoryOperation)
- (2.5.21.5 NAME 'attributeTypes'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'AttributeTypeDescription' USAGE directoryOperation)
- (2.5.21.6 NAME 'objectClasses'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'ObjectClassDescription' USAGE directoryOperation)

5.1.4. LDAP Operational Attributes

All servers must recognize the attribute types defined in this section.
(Of course, it is not required that the server provide values for these
attributes, when the attribute corresponds to a feature which the server
does not implement.)

- (1.3.6.1.4.1.1466.101.120.1 NAME 'administratorsAddress'
DESC 'This attribute's values are string containing the addresses of
the LDAP server's human administrator. This information may
be of use when tracking down problems in an Internet distributed
directory. For simplicity the syntax of the values are limited to
being URLs of the mailto form with an [RFC 822](#) address:
"mailto:user@domain". Future versions of this protocol may permit
other forms of addresses.'
SYNTAX 'IA5String' USAGE dSAOperation)

(1.3.6.1.4.1.1466.101.120.2 NAME 'currentTime'

DESC 'This attribute has a single value, a string containing a GeneralizedTime character string. This attribute need only be present if the server supports LDAP strong or protected simple authentication. Otherwise if the server does not know the current time, or does not choose to present it to clients, this attribute need not be present. The client may wish to use this value to detect whether a strong or protected bind is failing because the client and server clocks are not sufficiently synchronized. Clients must not use this time field for setting their own system clock.'

SYNTAX 'GeneralizedTime' SINGLE-VALUE USAGE dSAOperation)

(1.3.6.1.4.1.1466.101.120.3 NAME 'serverName'

DESC 'This attribute's value is the server's Distinguished Name. If the server does not have a Distinguished Name it will not be able to accept X.509-style strong authentication, and this attribute must be absent. However the presence of this attribute does not guarantee that the server will be able to perform strong authentication. If the server acts as a gateway to more than one X.500 DSA capable of strong authentication, there may be multiple values of this attribute, one per DSA. (Note: this attribute is distinct from myAccessPoint, for it is not required that a server have a presentation address in order to perform strong authentication.) (Note: it is likely that clients will retrieve this attribute in binary.)'

SYNTAX 'DN' USAGE dSAOperation)

(1.3.6.1.4.1.1466.101.120.4 NAME 'certificationPath'

DESC 'This attribute contains a binary DER encoding of an AF.CertificatePath data type, which is the certificate path for a server. If the server does not have a certificate path this attribute must be absent. (Note: this attribute may only be retrieved in binary.)'

SYNTAX 'CertificatePath' USAGE dSAOperation)

(1.3.6.1.4.1.1466.101.120.5 NAME 'namingContexts'

DESC 'The values of this attribute correspond to naming contexts which this server masters or shadows. If the server does not master any information (e.g. it is an LDAP gateway to a public X.500 directory) this attribute must be absent. If the server believes it contains the entire directory, the attribute must have a single value, and that value must be the empty string (indicating the null DN of the root). This attribute will allow clients to choose suitable base objects for searching when it has contacted a server.'

SYNTAX 'DN' USAGE dSAOperation)


```
( 1.3.6.1.4.1.1466.101.120.6 NAME 'altServer'
  DESC 'The values of this attribute are URLs of other servers which
        may be contacted when this server becomes unavailable.  If
        the server does not know of any other servers which could be
        used this attribute must be absent. Clients may cache this
        information in case their preferred LDAP server later becomes
        unavailable.'
  SYNTAX 'IA5String' USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.7 NAME 'supportedExtension'
  DESC 'The values of this attribute are OBJECT IDENTIFIERS,
        the names of supported extended operations
        which the server supports.  If the server does not support
        any extensions this attribute must be absent.'
  SYNTAX 'OID' USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.13 NAME 'supportedControl'
  DESC 'The values of this attribute are the names of supported session
        controls which the server supports.  If the server does not
        support any controls this attribute must be absent.'
  SYNTAX 'LDAPString' USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.14 NAME 'supportedSASLMechanisms'
  DESC 'The values of this attribute are the names of supported SASL
        mechanisms which the server supports.  If the server does not
        support any mechanisms this attribute must be absent.'
  SYNTAX 'LDAPString' USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.8 NAME 'entryName'
  SYNTAX 'DN' SINGLE-VALUE NO-USER-MODIFICATION USAGE directoryOperation )

( 1.3.6.1.4.1.1466.101.120.9 NAME 'modifyRights'
  SYNTAX 'ModifyRight' NO-USER-MODIFICATION USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.10 NAME 'incompleteEntry'
  SYNTAX 'BOOLEAN' NO-USER-MODIFICATION USAGE dSAOperation )

( 1.3.6.1.4.1.1466.101.120.11 NAME 'fromEntry'
  SYNTAX 'BOOLEAN' NO-USER-MODIFICATION USAGE dSAOperation )
```

5.1.5. LDAP User Attributes

The following attributes may be of use in naming entries, or as descriptive attributes in entries.

```
( 1.3.6.1.4.1.1466.101.121.1 NAME 'url'
  DESC 'Uniform Resource Locator'
  EQUALITY caseExactIA5Match SYNTAX 'IA5String' )
```

Note that the associatedDomain attribute may be used to hold a DNS name.

5.2. Syntaxes

5.2.1. Standard User Syntaxes

Servers must recognize all the syntaxes described in this section.

5.2.1.1. BitString

The encoding of a value with BitString syntax is according to the following BNF:

```
<bitstring> ::= ' ' <binary-digits> 'B'
```

```
<binary-digits> ::= '0' <binary-digits> | '1' <binary-digits> |  
empty
```

5.2.1.2. PrintableString

The encoding of a value with PrintableString syntax is the string value itself. PrintableString is limited to the characters in production <p> of [section 4.1](#).

5.2.1.3. DirectoryString

A string with DirectoryString syntax is encoded in the UTF-8 form of ISO 10646 (a superset of Unicode). Servers and clients must be prepared to receive arbitrary Unicode characters in values.

For characters in the PrintableString form, the value is encoded as the string value itself.

If it is of the TeletexString form, then the characters are transliterated to their equivalents in UniversalString, and encoded in UTF-8 [\[11\]](#).

If it is of the UniversalString or BMPString forms [\[12\]](#), UTF-8 is used to encode them.

Note: the form of DirectoryString is not indicated in protocol unless the attribute value is carried in binary. Servers which convert to DAP must choose an appropriate form. Servers must not reject values merely because they contain legal Unicode characters outside of the range of printable ASCII.

5.2.1.4. Certificate

Because of the changes from X.509(1988) and X.509(1993) and additional changes to the ASN.1 definition to support certificate extensions, no string representation is defined, and values with Certificate syntax must only be transferred using the binary encoding, by requesting or

returning the attributes with descriptions "userCertificate;binary" or "caCertificate;binary". The BNF notation in [RFC 1778](#) for "User Certificate" is not recommended to be used.

[5.2.1.5. CertificateList](#)

Because of the incompatibility of the X.509(1988) and X.509(1993) definitions of revocation lists, values with CertificateList syntax must only be transferred using a binary encoding, by requesting or returning the attributes with descriptions "certificateRevocationList;binary" or "authorityRevocationList;binary". The BNF notation in [RFC 1778](#) for "Authority Revocation List" is not recommended to be used.

[5.2.1.6. CertificatePair](#)

Because the Certificate is being carried in binary, values with CertificatePair syntax must only be transferred using a binary encoding, by requesting or returning the attribute description "crossCertificatePair;binary". The BNF notation in [RFC 1778](#) for "Certificate Pair" is not recommended to be used.

[5.2.1.7. CountryString](#)

A value of CountryString syntax is encoded the same as a value of DirectoryString syntax. Note that this syntax is limited to values of exactly two printable string characters.

```
<CountryString> ::= <p> <p>
```

[5.2.1.8. DN](#)

Values with DN (Distinguished Name) syntax are encoded to have the representation defined in [5]. Note that this representation is not reversible to the original ASN.1 encoding as the CHOICE of any DirectoryString element in an RDN is no longer known.

[5.2.1.9. DeliveryMethod](#)

Values with DeliveryMethod syntax are encoded according to the following BNF:

```
<delivery-value> ::= <pdm> | <pdm> '$' <delivery-value>

<pdm> ::= 'any' | 'mhs' | 'physical' | 'telex' | 'teletex' |
          'g3fax' | 'g4fax' | 'ia5' | 'videotex' | 'telephone'
```

[5.2.1.10. EnhancedGuide](#)

Values with the EnhancedGuide syntax are encoded according to the following BNF:

```
<EnhancedGuide> ::= <objectclass> '#' <criteria> '#' <subset>
```

```
<subset> ::= "baseobject" | "oneLevel" | "wholeSubtree"
```

The <criteria> production is defined in the Guide syntax below.
This syntax has been added subsequent to [RFC 1778](#).

[5.2.1.11](#). FacsimileTelephoneNumber

Values with the FacsimileTelephoneNumber syntax are encoded according to the following BNF:

```
<fax-number> ::= <printablestring> [ '$' <faxparameters> ]  
  
<faxparameters> ::= <faxparm> | <faxparm> '$' <faxparameters>  
  
<faxparm> ::= 'twoDimensional' | 'fineResolution' | 'unlimitedLength' |  
              'b4Length' | 'a3Width' | 'b4Width' | 'uncompressed'
```

In the above, the first <printablestring> is the actual fax number, and the <faxparm> tokens represent fax parameters.

[5.2.1.12](#). Guide

Values with the Guide syntax are encoded according to the following BNF:

```
<guide-value> ::= [ <object-class> '#' ] <criteria>  
  
<object-class> ::= an encoded value with OID syntax  
  
<criteria> ::= <criteria-item> | <criteria-set> | '!' <criteria>  
  
<criteria-set> ::= [ '(' ] <criteria> '&' <criteria-set> [ ')' ] |  
                  [ '(' ] <criteria> '|' <criteria-set> [ ')' ]  
  
<criteria-item> ::= [ '(' ] <attributetype> '$' <match-type> [ ')' ]  
  
<match-type> ::= "EQ" | "SUBSTR" | "GE" | "LE" | "APPROX"
```

[5.2.1.13](#). NameAndOptionalUID

The encoding of a value with the NameAndOptionalUID syntax is according to the following BNF:

```
<NameAndOptionalUID> ::=  
    <DistinguishedName> [ '#' <BitString> ]
```

Although the '#' character may occur in a string representation of a distinguished name, no additional special quoting is done in the distinguished name other than that of [\[5\]](#).

This syntax has been added subsequent to [RFC 1778](#).

[5.2.1.14](#). NumericString

The encoding of a string with the NumericString syntax is the string value itself.

[5.2.1.15](#). OID

Values with OID (Object Identifier) syntax are encoded according to the following BNF:

```
<oid> ::= <descr> | <numericoid>
```

```
<descr> ::= <keystring>
```

```
<numericoid> ::= <numericstring> | <numericstring> '.' <numericoid>
```

In the above BNF, <descr> is the syntactic representation of an object descriptor, which must consist of letters and digits, starting with a letter. When encoding values with OID syntax, the first encoding option must be used in preference to the second. That is, in encoding object identifiers, object descriptors (where assigned and known by the implementation) must be used in preference to numeric oids to the greatest extent possible. All permitted object descriptors for use in LDAP are given in this document. No other object descriptors may be used. (Note that clients must expect that LDAPv2 implementations will return object descriptors other than those listed.)

[5.2.1.16](#). Password

Values with Password syntax are encoded as octet strings.

[5.2.1.17](#). PostalAddress

Values with the PostalAddress syntax are encoded according to the following BNF:

```
<postal-address> ::= <dstring> | <dstring> '$' <postal-address>
```

In the above, each <dstring> component of a postal address value is encoded as a value of type DirectoryString syntax. Backslashes and dollar characters, if they occur in the component, are quoted as described in [section 4.2](#).

[5.2.1.18](#). PresentationAddress

Values with the PresentationAddress syntax are encoded to have the representation described in [\[6\]](#).

[5.2.1.20](#). TelephoneNumber

Values with the TelephoneNumber syntax are encoded as if they were Printable String types. Telephone numbers are recommended in X.520 to be in international form, e.g. "+1 512 305 0280".

5.2.1.21. TeletexTerminalIdentifier

Values with the TeletexTerminalIdentifier syntax are encoded according to the following BNF:

```
<teletex-id> ::= <ttx-term> 0*('$' <ttx-param>)  
  
<ttx-term> ::= <printablestring>  
  
<ttx-param> ::= <ttx-key> ':' <ttx-value>  
  
<ttx-key> ::= 'graphic' | 'control' | 'misc' | 'page' | 'private'  
  
<ttx-value> ::= <octetstring>
```

In the above, the first <printablestring> is the encoding of the first portion of the teletex terminal identifier to be encoded, and the subsequent 0 or more <octetstrings> are subsequent portions of the teletex terminal identifier.

5.2.1.22. TelexNumber

Values with the TelexNumber syntax are encoded according to the following BNF:

```
<telex-number> ::= <actual-number> '$' <country> '$' <answerback>  
  
<actual-number> ::= <printablestring>  
  
<country> ::= <printablestring>  
  
<answerback> ::= <printablestring>
```

In the above, <actual-number> is the syntactic representation of the number portion of the TELEX number being encoded, <country> is the TELEX country code, and <answerback> is the answerback code of a TELEX terminal.

5.2.1.23. UTCTime

Values with UTCTime syntax are encoded as if they were printable strings with the strings containing a UTCTime value. This is historical; new attribute definitions must use GeneralizedTime instead.

5.2.1.24. Boolean

Values with Boolean syntax are encoded according to the following BNF:

```
<boolean> ::= "TRUE" | "FALSE"
```

Boolean values have an encoding of "TRUE" if they are logically true,
and have an encoding of "FALSE" otherwise.

5.2.2. Pilot Syntaxes

Servers must recognize all the syntaxes described in this section.

5.2.2.1. Audio

The encoding of a value with Audio syntax is the octets of the value itself, an 8KHz uncompressed encoding compatible with the SunOS 4.1.3 'play' utility.

5.2.2.2. DSAQualitySyntax

Values with this syntax are encoded according to the following BNF:

```
<DsaQualitySyntax> ::= <DSAKeyword> [ '#' <description> ]  
  
<DSAKeyword> ::= 'DEFUNCT' | 'EXPERIMENTAL' | 'BEST-EFFORT' |  
                  'PILOT-SERVICE' | 'FULL-SERVICE'  
  
<description> ::= encoded as a PrintableString
```

5.2.2.3. DataQualitySyntax

Values with this syntax are encoded according to the following BNF:

```
<DataQualitySyntax> ::= <compKeyword> '#' <attrQuality> '#'  
                        <listQuality> [ '#' <description> ]  
  
<attrQuality> ::= <levelKeyword> '+' <compKeyword>  
  
<listQuality> ::= <list> '$' <list><listQuality>  
  
<list> ::= <attribute> '+' <attrQuality>  
  
<compKeyword> ::= 'NONE' | 'SAMPLE' | 'SELECTED' |  
                  'SUBSTANTIAL' | 'FULL'  
  
<levelKeyword> ::= 'UNKNOWN' | 'EXTERNAL' | 'SYSTEM-MAINTAINED' |  
                  'USER-SUPPLIED'
```

5.2.2.4. IA5String

The encoding of a value with IA5String syntax is the string value itself.

5.2.2.5. JPEG

Values with JPEG syntax are encoded as if they were octet strings containing JPEG images in the JPEG File Interchange Format (JFIF), as described in [8].

5.2.2.6. MailPreference

Values with MailPreference syntax are encoded according to the following BNF:

```
<mail-preference> ::= "NO-LISTS" | "ANY-LIST" | "PROFESSIONAL-LISTS"
```

5.2.2.7. OtherMailbox

Values of the OtherMailbox syntax are encoded according to the following BNF:

```
<otherMailbox> ::= <mailbox-type> '$' <mailbox>
```

```
<mailbox-type> ::= an encoded Printable String
```

```
<mailbox> ::= an encoded IA5 String
```

In the above, <mailbox-type> represents the type of mail system in which the mailbox resides, for example "MCIMail"; and <mailbox> is the actual mailbox in the mail system defined by <mailbox-type>.

5.2.2.8. Fax

Values with Fax syntax are encoded as if they were octet strings containing Group 3 Fax images as defined in [7].

5.2.3. Operational Syntaxes

Servers must recognize all the syntaxes described in this section.

5.2.3.1. AttributeTypeDescription

Values with this syntax are encoded according to the BNF given at the start of [section 4.1](#). For example,

```
( 2.5.4.0 NAME 'objectClass' SYNTAX 'OID' )
```

5.2.3.2. GeneralizedTime

Values of this syntax are encoded as printable strings, represented as specified in X.208. Note that the time zone must be specified. It is strongly recommended that Zulu time zone be used. For example,

```
199412161032Z
```

5.2.3.3. INTEGER

Values with INTEGER syntax are encoded as the decimal representation of their values, with each decimal digit represented by the its character equivalent. So the number 1321 is represented by the character

```
string "1321".
```

5.2.3.4. ObjectClassDescription

Values of this syntax are encoded according to the BNF in [section 4.3](#).

5.3. Object Classes

5.3.1. Standard Classes

Servers must recognize the object classes listed here as values of the objectClass attribute. With the exception of groupOfUniqueNames, they are described in [RFC 1274](#).

```
( 2.5.6.0 NAME 'top' ABSTRACT MUST objectClass )

( 2.5.6.1 NAME 'alias' SUP top STRUCTURAL MUST aliasedObjectName )

( 2.5.6.2 NAME 'country' SUP top STRUCTURAL MUST c
  MAY ( searchGuide $ description ) )

( 2.5.6.3 NAME 'locality' SUP top STRUCTURAL
  MAY ( street $ seeAlso $ searchGuide $ st $ l $ description ) )

( 2.5.6.4 NAME 'organization' SUP top STRUCTURAL MUST o
  MAY ( userPassword $ searchGuide $ seeAlso $ businessCategory $
    x121Address $ registeredAddress $ destinationIndicator $
    preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
    telephoneNumber $ internationalISDNNumber $ facsimileTelephoneNumber $
    street $ postOfficeBox $ postalCode $ postalAddress $
    physicalDeliveryOfficeName $ st $ l $ description ) )

( 2.5.6.5 NAME 'organizationalUnit' SUP top STRUCTURAL MUST ou
  MAY ( userPassword $ searchGuide $ seeAlso $ businessCategory $
    x121Address $ registeredAddress $ destinationIndicator $
    preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
    telephoneNumber $ internationalISDNNumber $ facsimileTelephoneNumber $
    street $ postOfficeBox $ postalCode $ postalAddress $
    physicalDeliveryOfficeName $ st $ l $ description ) )

( 2.5.6.6 NAME 'person' SUP top STRUCTURAL MUST ( sn $ cn )
  MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )

( 2.5.6.7 NAME 'organizationalPerson' SUP person STRUCTURAL
  MAY ( title $ x121Address $ registeredAddress $ destinationIndicator $
    preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
    telephoneNumber $ internationalISDNNumber $ facsimileTelephoneNumber $
    street $ postOfficeBox $ postalCode $ postalAddress $
    physicalDeliveryOfficeName $ ou $ st $ l ) )
```


- (2.5.6.8 NAME 'organizationalRole' SUP top STRUCTURAL MUST cn
MAY (x121Address \$ registeredAddress \$ destinationIndicator \$
preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$
telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$
seeAlso \$ roleOccupant \$ preferredDeliveryMethod \$ street \$
postOfficeBox \$ postalCode \$ postalAddress \$
physicalDeliveryOfficeName \$ ou \$ st \$ l \$ description))
- (2.5.6.9 NAME 'groupOfNames' SUP top STRUCTURAL MUST (member \$ cn)
MAY (businessCategory \$ seeAlso \$ owner \$ ou \$ o \$ description))
- (2.5.6.10 NAME 'residentialPerson' SUP person STRUCTURAL MUST 1
MAY (businessCategory \$ x121Address \$ registeredAddress \$
destinationIndicator \$ preferredDeliveryMethod \$ telexNumber \$
teletexTerminalIdentifier \$ telephoneNumber \$ internationaliSDNNumber \$
facsimileTelephoneNumber \$ preferredDeliveryMethod \$ street \$
postOfficeBox \$ postalCode \$ postalAddress \$
physicalDeliveryOfficeName \$ st \$ l))
- (2.5.6.11 NAME 'applicationProcess' SUP top STRUCTURAL MUST cn
MAY (seeAlso \$ ou \$ l \$ description))
- (2.5.6.12 NAME 'applicationEntity' SUP top STRUCTURAL
MUST (presentationAddress \$ cn)
MAY (supportedApplicationContext \$ seeAlso \$ ou \$ o \$ l \$
description))
- (2.5.6.13 NAME 'dSA' SUP applicationEntity STRUCTURAL
MAY knowledgeInformation)
- (2.5.6.14 NAME 'device' SUP top STRUCTURAL MUST cn
MAY (serialNumber \$ seeAlso \$ owner \$ ou \$ o \$ l \$ description))
- (2.5.6.15 NAME 'strongAuthenticationUser' SUP top STRUCTURAL
MUST userCertificate)
- (2.5.6.16 NAME 'certificationAuthority' SUP top STRUCTURAL
MUST (authorityRevocationList \$ certificateRevocationList \$
cACertificate) MAY crossCertificatePair)
- (2.5.6.17 NAME 'groupOfUniqueNames' SUP top STRUCTURAL
MUST (uniqueMember \$ cn)
MAY (businessCategory \$ seeAlso \$ owner \$ ou \$ o \$ description))

5.3.2. Pilot Classes

These object classes are defined in [RFC 1274](#). All servers must recognize these object class names.

- (0.9.2342.19200300.100.4.3 NAME 'pilotObject' SUP top STRUCTURAL

```
MAY ( jpegPhoto $ audio $ dITRedirect $ lastModifiedBy $  
lastModifiedTime $ uniqueIdentifier $ manager $ photo $ info ) )
```


- (0.9.2342.19200300.100.4.4 NAME 'newPilotPerson' SUP person
STRUCTURAL MAY (personalSignature \$ mailPreferenceOption \$
organizationalStatus \$ pagerTelephoneNumber \$ mobileTelephoneNumber \$
otherMailbox \$ janetMailbox \$ businessCategory \$
preferredDeliveryMethod \$ personalTitle \$ secretary \$
homePostalAddress \$ homePhone \$ userClass \$ roomNumber \$
favouriteDrink \$ rfc822Mailbox \$ textEncodedORaddress \$ userid))
- (0.9.2342.19200300.100.4.5 NAME 'account' SUP top STRUCTURAL
MUST userid MAY (host \$ ou \$ o \$ l \$ seeAlso \$ description))
- (0.9.2342.19200300.100.4.6 NAME 'document' SUP (top \$ pilotObject)
STRUCTURAL MUST documentIdentifier
MAY (documentPublisher \$ documentStore \$ documentAuthorSurName \$
documentAuthorCommonName \$ abstract \$ subject \$ keywords \$
updatedByDocument \$ updatesDocument \$ obsoletedByDocument \$
obsoletesDocument \$ documentLocation \$ documentAuthor \$
documentVersion \$ documentTitle \$ ou \$ o \$ l \$ seeAlso \$ description \$
cn))
- (0.9.2342.19200300.100.4.7 NAME 'room' SUP top STRUCTURAL MUST cn
MAY (telephoneNumber \$ seeAlso \$ description \$ roomNumber))
- (0.9.2342.19200300.100.4.9 NAME 'documentSeries' SUP top STRUCTURAL
MUST cn MAY (ou \$ o \$ l \$ telephoneNumber \$ seeAlso \$ description))
- (0.9.2342.19200300.100.4.13 NAME 'domain' SUP top STRUCTURAL
MUST dc
MAY (userPassword \$ searchGuide \$ seeAlso \$ businessCategory \$
x121Address \$ registeredAddress \$ destinationIndicator \$
preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$
telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$
street \$ postOfficeBox \$ postalCode \$ postalAddress \$
physicalDeliveryOfficeName \$ st \$ l \$ description \$ o \$
associatedName))
- (0.9.2342.19200300.100.4.14 NAME 'rFC822localPart' SUP domain
STRUCTURAL
MAY (x121Address \$ registeredAddress \$ destinationIndicator \$
preferredDeliveryMethod \$ telexNumber \$ teletexTerminalIdentifier \$
telephoneNumber \$ internationaliSDNNumber \$ facsimileTelephoneNumber \$
streetAddress \$ postOfficeBox \$ postalCode \$ postalAddress \$
physicalDeliveryOfficeName \$ telephoneNumber \$ seeAlso \$ description \$
sn \$ cn))
- (0.9.2342.19200300.100.4.15 NAME 'DNSDomain' SUP domain STRUCTURAL
MAY DNSRecord)
- (0.9.2342.19200300.100.4.17 NAME 'domainRelatedObject' SUP top

STRUCTURAL MUST associatedDomain)

(0.9.2342.19200300.100.4.18 NAME 'friendlyCountry' SUP country
STRUCTURAL MUST co)

```
( 0.9.2342.19200300.100.4.19 NAME 'simpleSecurityObject' SUP top
  STRUCTURAL MUST userPassword )

( 0.9.2342.19200300.100.4.20 NAME 'pilotOrganization'
  SUP ( organization $ organizationalUnit ) STRUCTURAL
  MAY buildingName )

( 0.9.2342.19200300.100.4.21 NAME 'pilotDSA' SUP dSA STRUCTURAL
  MUST dSAQuality )

( 0.9.2342.19200300.100.4.23 NAME 'qualityLabelledData' SUP top
  STRUCTURAL MUST singleLevelQuality
  MAY ( subtreeMaximumQuality $ subtreeMinimumQuality ) )
```

5.4. Matching Rules

Servers must recognize the following matching rules, used for equality matching, and must be capable of performing the matching rules. For all these rules, the assertion syntax is the same as the value syntax.

```
( 2.5.13.0 NAME 'objectIdentifierMatch' SYNTAX 'OID' )
( 2.5.13.1 NAME 'distinguishedNameMatch' SYNTAX 'DN' )
( 2.5.13.2 NAME 'caseIgnoreMatch' SYNTAX 'DirectoryString' )
( 2.5.13.8 NAME 'numericStringMatch' SYNTAX 'NumericString' )
( 2.5.13.11 NAME 'caseIgnoreListMatch' SYNTAX 'PostalAddress' )
( 2.5.13.14 NAME 'integerMatch' SYNTAX 'INTEGER' )
( 2.5.13.16 NAME 'bitStringMatch' SYNTAX 'BitString' )
( 2.5.13.17 NAME 'octetStringMatch' SYNTAX 'Password' )
( 2.5.13.20 NAME 'telephoneNumberMatch' SYNTAX 'TelephoneNumber' )
( 2.5.13.27 NAME 'generalizedTimeMatch' SYNTAX 'GeneralizedTime' )
( 1.3.6.1.4.1.1466.109.114.1 NAME 'caseExactIA5Match' SYNTAX 'IA5String' )
( 1.3.6.1.4.1.1466.109.114.2 NAME 'caseIgnoreIA5Match' SYNTAX 'IA5String' )
```

When performing the caseIgnoreMatch, caseIgnoreListMatch, telephoneNumberMatch, caseExactIA5Match and caseIgnoreIA5Match, multiple adjoining whitespace characters are treated the same as an individual space, and leading and trailing whitespace is ignored.

6. X.500 Definitions

Servers which implement the X.500(1993) protocols are required to recognize these attributes types, syntaxes, object classes and matching rules, where they correspond to X.500 features implemented by that server. No other servers are required to implement any definitions in [section 6](#), although they may do so.

Clients must not assume these definitions are recognized by all servers.

6.1. Attribute Types

6.1.1. User Attributes

All user attributes of X.500 are listed in [section 5.1.1](#).

6.1.2. Collective Attributes

These attributes are stored in collective attribute subentries, but may be visible in user entries if requested.

Each of these collective attributes is a subtype of the attribute which has the OID without the final ".1", e.g. "collectivePostalCode" is a subtype of "postalCode".

```
( 2.5.4.7.1 NAME 'collectiveLocalityName' SUP l COLLECTIVE )  
  
( 2.5.4.8.1 NAME 'collectiveStateOrProvinceName' SUP st COLLECTIVE )  
  
( 2.5.4.9.1 NAME 'collectiveStreetAddress' SUP street COLLECTIVE )  
  
( 2.5.4.10.1 NAME 'collectiveOrganizationName' SUP o COLLECTIVE )  
  
( 2.5.4.11.1 NAME 'collectiveOrganizationalUnitName' SUP ou COLLECTIVE )  
  
( 2.5.4.16.1 NAME 'collectivePostalAddress' SUP postalAddress COLLECTIVE )  
  
( 2.5.4.17.1 NAME 'collectivePostalCode' SUP postalCode COLLECTIVE )  
  
( 2.5.4.18.1 NAME 'collectivePostOfficeBox' SUP postOfficeBox COLLECTIVE )  
  
( 2.5.4.19.1 NAME 'collectivePhysicalDeliveryOfficeName'  
  SUP physicalDeliveryOfficeName COLLECTIVE )  
  
( 2.5.4.20.1 NAME 'collectiveTelephoneNumber' SUP telephoneNumber  
  COLLECTIVE )  
  
( 2.5.4.21.1 NAME 'collectiveTelexNumber' SUP 'TelexNumber' COLLECTIVE )  
  
( 2.5.4.22.1 NAME 'collectiveTeletexTerminalIdentifier'  
  SUP teletexTerminalIdentifier COLLECTIVE )  
  
( 2.5.4.23.1 NAME 'collectiveFacsimileTelephoneNumber'  
  SUP facsimileTelephoneNumber COLLECTIVE )  
  
( 2.5.4.25.1 NAME 'collectiveInternationaliSDNNumber'  
  SUP internationaliSDNNumber COLLECTIVE )
```

6.1.3. Standard Operational Attributes

These attributes are defined in X.501(1993) Annexes B through E.

```
( 2.5.18.5 NAME 'administrativeRole' EQUALITY objectIdentifierMatch  
  SYNTAX 'OID' USAGE directoryOperation )  
  
( 2.5.18.6 NAME 'subtreeSpecification' SYNTAX 'SubtreeSpecification'  
  SINGLE-VALUE USAGE directoryOperation )
```


- (2.5.18.7 NAME 'collectiveExclusions' EQUALITY objectIdentifierMatch
SYNTAX 'OID' USAGE directoryOperation)
- (2.5.21.1 NAME 'dITStructureRules' EQUALITY integerFirstComponentMatch
SYNTAX 'DITStructureRuleDescription' USAGE directoryOperation)
- (2.5.21.2 NAME 'dITContentRules'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'DITContentRuleDescription' USAGE directoryOperation)
- (2.5.21.4 NAME 'matchingRules'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'MatchingRuleDescription' USAGE directoryOperation)
- (2.5.21.7 NAME 'nameForms'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'NameFormDescription' USAGE directoryOperation)
- (2.5.21.8 NAME 'matchingRuleUse'
EQUALITY objectIdentifierFirstComponentMatch
SYNTAX 'MatchingRuleUseDescription' USAGE directoryOperation)
- (2.5.21.9 NAME 'structuralObjectClass' EQUALITY objectIdentifierMatch
SYNTAX 'OID' SINGLE-VALUE NO-USER-MODIFICATION
USAGE directoryOperation)
- (2.5.21.10 NAME 'governingStructuralRule' EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE NO-USER-MODIFICATION
USAGE directoryOperation)
- (2.5.24.1 NAME 'accessControlScheme' EQUALITY objectIdentifierMatch
SYNTAX 'OID' SINGLE-VALUE USAGE directoryOperation)
- (2.5.24.4 NAME 'prescriptiveACI'
EQUALITY directoryStringFirstComponentMatch SYNTAX 'ACIItem'
USAGE directoryOperation)
- (2.5.24.5 NAME 'entryACI'
EQUALITY directoryStringFirstComponentMatch SYNTAX 'ACIItem'
USAGE directoryOperation)
- (2.5.24.6 NAME 'subentryACI'
EQUALITY directoryStringFirstComponentMatch SYNTAX 'ACIItem'
USAGE directoryOperation)
- (2.5.12.0 NAME 'dseType' EQUALITY bitStringMatch SYNTAX 'DSEType'
SINGLE-VALUE NO-USER-MODIFICATION USAGE dSAOperation)
- (2.5.12.1 NAME 'myAccessPoint' EQUALITY accessPointMatch
SYNTAX 'AccessPoint' SINGLE-VALUE NO-USER-MODIFICATION

USAGE dSAOperation)

Wahl, Coulbeck, Howes & Kille

[Page 28]


```
( 2.5.12.2 NAME 'superiorKnowledge' EQUALITY accessPointMatch
  SYNTAX 'AccessPoint' SINGLE-VALUE NO-USER-MODIFICATION
  USAGE dSAOperation )

( 2.5.12.3 NAME 'specificKnowledge'
  EQUALITY masterAndShadowAccessPointsMatch
  SYNTAX 'MasterAndShadowAccessPoints'
  SINGLE-VALUE NO-USER-MODIFICATION USAGE distributedOperation )

( 2.5.12.4 NAME 'nonSpecificKnowledge'
  EQUALITY masterAndShadowAccessPointsMatch
  SYNTAX 'MasterAndShadowAccessPoints' NO-USER-MODIFICATION
  USAGE distributedOperation )

( 2.5.12.5 NAME 'supplierKnowledge'
  EQUALITY supplierOrConsumerInformationMatch
  SYNTAX 'SupplierInformation'
  NO-USER-MODIFICATION USAGE dSAOperation )

( 2.5.12.6 NAME 'consumerKnowledge'
  EQUALITY supplierOrConsumerInformationMatch
  SYNTAX 'SupplierOrConsumer'
  NO-USER-MODIFICATION USAGE dSAOperation )

( 2.5.12.7 NAME 'secondaryShadows'
  EQUALITY supplierAndConsumersMatch
  SYNTAX 'SupplierAndConsumers'
  NO-USER-MODIFICATION USAGE dSAOperation )
```

6.1.4. LDAP-defined Operational Attributes

6.1.4.1. targetSystem

```
( 1.3.6.1.4.1.1466.101.120.12 NAME 'targetSystem'
  SYNTAX 'AccessPoint' SINGLE-VALUE NO-USER-MODIFICATION
  USAGE distributedOperation )
```

The value of this attribute may be supplied in an AddEntry operation to inform the Directory of the target server on which the entry is to be held. This is used to create a new naming context in the directory tree. A server which does not permit the use of this attribute must return an appropriate error code if it is present in the attribute list. This attribute will generally not be present in the entry after the add is completed.

6.2. Syntaxes

6.2.1. Standard Syntaxes

6.2.1.1. ACIItem

This syntax appears too complicated for a compact string representation to be useful. Clients must only request and servers must only return values which use the the binary encoding of the value, e.g. "entryACI;binary".

It is recommended that clients that wish to only determine whether they have been granted permission to modify an entry use the "modifyRights" attribute rather than attempt to parse this syntax.

6.2.1.2. AccessPoint

Values with AccessPoint syntax are encoded according to the following BNF:

```
<AccessPoint> ::= ( '(' <DistinguishedName> '#'
                    <PresentationAddress> ')' ) |
    -- Optional protocol info absent, parenthesis required
    ( '(' <DistinguishedName> '#'
      <PresentationAddress> '#'
      <SetOfProtocolInformation> ')' )

<SetOfProtocolInformation> ::= <ProtocolInformation> |
    '(' <ProtocolInformationList> ')'

<ProtocolInformationList> ::= <ProtocolInformation> |
    <ProtocolInformation> '$'
    <ProtocolInformationList>
```

6.2.1.3. DITContentRuleDescription

Values with this syntax are encoded according to the following BNF:

```
<DITContentRuleDescription> ::= "("
    <oid>    -- Structural ObjectClass identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    [ "AUX" <oids> ]      -- Auxiliary ObjectClasses
    [ "MUST" <oids> ]     -- AttributeType identifiers
    [ "MAY" <oids> ]      -- AttributeType identifiers
    [ "NOT" <oids> ]       -- AttributeType identifiers
    ")"
```


6.2.1.4. DITStructureRuleDescription

Values with this syntax are encoded according to the following BNF:

```

<DITStructureRuleDescription> ::= "("
    <RuleIdentifier>      -- DITStructureRule identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    "FORM" <oid>          -- NameForm
    [ "SUP" <RuleIdentifiers> ] -- superior DITStructureRules
    ")"

<RuleIdentifier> ::= <integer>

<RuleIdentifiers> ::=
    <RuleIdentifier> |
    "(" <RuleIdentifierList> ")"

<RuleIdentifierList> ::=
    <RuleIdentifierList> <RuleIdentifier>
|
    -- empty list

```

6.2.1.5. DSEType

Values with DSEType syntax are encoded according to the following BNF:

```

<DSEType> ::= '(' <DSEBitList> ')'

<DSEBitList> ::= <DSEBit> | <DSEBit> '$' <DSEBitList>

<DSEBit> ::= 'root' | 'glue' | 'cp' | 'entry' | 'alias' | 'subr' |
    'nssr' | 'supr' | 'xr' | 'admPoint' | 'subentry' |
    'shadow' | 'zombie' | 'immSupr' | 'rhob' | 'sa'

```

6.2.1.6. MasterAndShadowAccessPoints

Values of this syntax are encoded according to the following BNF:

```

<MasterAndShadowAccessPoints> ::= <MasterOrShadowAccessPoint> |
    '(' <MasterAndShadowAccessPointList> ')'

<MasterAndShadowAccessPointList> ::= <MasterOrShadowAccessPoint> |
    <MasterOrShadowAccessPoint> '$' <MasterAndShadowAccessPointList>

<MasterOrShadowAccessPoint> ::= <category> '#' <AccessPoint>

<category> ::= 'master' | 'shadow'

```

6.2.1.7. MatchingRuleDescription

Values of this syntax are encoded according to the BNF of [section 4.4.](#)

6.2.1.8. MatchingRuleUseDescription

Values of this syntax are encoded according to the following BNF:

```
<MatchingRuleUseDescription> ::= "("
    <oid>    -- MatchingRule identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    "APPLIES" <oids>    -- AttributeType identifiers
    ")"
```

6.2.1.9. NameFormDescription

Values of this syntax are encoded according to the following BNF:

```
<NameFormDescription> ::= "("
    <oid>    -- NameForm identifier
    [ "NAME" <DirectoryStrings> ]
    [ "DESC" <DirectoryString> ]
    [ "OBSOLETE" ]
    "OC" <oid>    -- Structural ObjectClass
    "MUST" <oids>    -- AttributeTypes
    [ "MAY" <oids> ]    -- AttributeTypes
    ")"
```

6.2.1.10. SubtreeSpecification

Values of this syntax are encoded according to the following BNF:

```
<SubtreeSpecification> ::= '(' [<localname>] '#'
                                [<exclusionlist>] '#'
                                [<minimum>] '#' [<maximum>] '#'
                                [<refinement>] ')'

<localname> ::= <DistinguishedName>

<exclusionlist> ::= '(' <exclusions> ')'

<exclusions> ::= <exclusion> | <exclusion> '$' <exclusionlist>

<exclusion> ::= ( 'before ' <DistinguishedName> ) |
               ( 'after ' <DistinguishedName> )

<minimum> ::= <numericstring>

<maximum> ::= <numericstring>

<refinement> ::= <oid> | '!' <refinement> |
                 '( &' <refinements> ')'
```

'(|' <refinements> ')'

<refinements> ::= <refinement> | <refinement> '\$' <refinements>

6.2.1.11. SupplierInformation

Values of this syntax are encoded according to the following BNF:

```
<SupplierInformation> ::=
  -- supplier is master --
  '(' 'master' '#' <SupplierOrConsumer> ')' |

  -- supplier is not master, master unspecified --
  '(' 'shadow' '#' <SupplierOrConsumer> ')' |

  -- supplier not master, master specified --
  '[' '(' 'shadow' '#' <SupplierOrConsumer> '#' <AccessPoint> [')'] ]
```

6.2.1.12. SupplierOrConsumer

Values of this syntax are encoded according to the following BNF:

```
<SupplierOrConsumer> ::= <Agreement> '#' <AccessPoint>

<Agreement> ::= <bindingid> '.' <bindingversion>

<bindingid> ::= <numericstring>

<bindingversion> ::= <numericstring>
```

6.2.1.13. SupplierAndConsumers

Values of this syntax are encoded according to the following BNF:

```
<SupplierAndConsumers> ::= <Supplier> '#' <Consumers>

<Suppliers> ::= <AccessPoint>

<Consumers> ::= <AccessPoint> | '(' <AccessPointList> ')'

<AccessPointList> ::= <AccessPoint> |
  <AccessPoint> '$' <AccessPointList>
```

6.2.1.14. ProtocolInformation

A value with the ProtocolInformation syntax is encoded according to the following BNF:

```
<ProtocolInformation> ::= <NetworkAddress> <space> '#'
  <SetOfProtocolIdentifier>

<NetworkAddress> ::= As appears in PresentationAddress

<SetOfProtocolIdentifiers> ::= <ProtocolIdentifier> |
  '(' <ProtocolIdentifiers> ')'
```



```

<ProtocolIdentifiers> ::= <ProtocolIdentifier> |
                        <ProtocolIdentifier> '$' <ProtocolIdentifiers>

<ProtocolIdentifier> ::= <oid>

```

For example,

```
NS+12345678 # 1.2.3.4.5
```

[6.2.2. LDAP-defined Syntaxes](#)

There is currently one syntax defined here.

[6.2.2.1 ModifyRight](#)

This syntax is a printable encoding of the following ASN.1 data type:

```

ModifyRight ::= SEQUENCE {
  item CHOICE {
    entry      [0] NULL,
    attribute  [1] AttributeType,
    value      [2] AttributeValueAssertion },
  permission  [3] BIT STRING { add(0), remove(1), rename(2), move(3) } }

```

The syntax is encoded according to the following BNF:

```

<ModifyRight> ::= [<perm-list>] <octo> <item>
  -- perm list is absent when none of the bits set in permission

<item> ::= <entry> | <attribute> | <value>

<entry> ::= 'entry'

<attribute> ::= 'attribute' <dollar> <attributetype>

<value> ::= 'value' <dollar> <attributetype> <dollar> <strvalue>

-- <strvalue> is the string encoding of the value

<perm-list> ::= <perm> | <perm> <dollar> <perm-list>
  -- one or more of the bits in permission, if set

<perm> ::= 'add' | 'remove' | 'rename' | 'move'

<octo> ::= [ <whsp> ] '#' [ <whsp> ]

<dollar> ::= [ <whsp> ] '$' [ <whsp> ]

```


For example,

```
# entry
add $ remove # attribute $ cn
add $ remove # attribute $ sn
remove # value $ memberName $ CN=Babs, O=Michigan, C=US
```

6.3. Object Classes

The following object classes may be recognized.

```
( 2.5.17.0 NAME 'subentry' SUP top STRUCTURAL
  MUST ( cn $ subtreeSpecification ) )

( 2.5.17.1 NAME 'accessControlSubentry' AUXILIARY )

( 2.5.17.2 NAME 'collectiveAttributeSubentry' AUXILIARY )

( 2.5.20.1 NAME 'subschema' AUXILIARY
  MAY ( dITStructureRules $ nameForms $ ditContentRules $
    objectClasses $ attributeTypes $ matchingRules $ matchingRuleUse ) )
```

6.4. Matching Rules

Only servers which implement the attribute types which reference these matching rules in their definition are required to implement these rules.

The definitions of the rules can be found in [\[2\]](#) and [\[3\]](#).

Name	OID
=====	=====
caseIgnoreOrderingMatch	2.5.13.3
caseIgnoreSubstringsMatch	2.5.13.4
caseExactMatch	2.5.13.5
caseExactOrderingMatch	2.5.13.6
caseExactSubstringsMatch	2.5.13.7
numericStringOrderingMatch	2.5.13.9
numericStringSubstringsMatch	2.5.13.10
caseIgnoreListSubstringsMatch	2.5.13.12
booleanMatch	2.5.13.13
integerOrderingMatch	2.5.13.15
octetStringOrderingMatch	2.5.13.18
octetStringSubstringsMatch	2.5.13.19
telephoneNumberSubstringsMatch	2.5.13.21
presentationAddressMatch	2.5.13.22
uniqueMemberMatch	2.5.13.23
protocolInformationMatch	2.5.13.24
uTCTimeMatch	2.5.13.25
uTCTimeOrderingMatch	2.5.13.26
generalizedTimeOrderingMatch	2.5.13.28

integerFirstComponentMatch	2.5.13.29
objectIdentifierFirstComponentMatch	2.5.13.30
directoryStringFirstComponentMatch	2.5.13.31

wordMatch	2.5.13.32
keywordMatch	2.5.13.33
accessPointMatch	2.5.14.0
masterAndShadowAccessPointsMatch	2.5.14.1
supplierOrConsumerInformationMatch	2.5.14.2
supplierAndConsumersMatch	2.5.14.3

6.5. Other

The string 'excludeAllCollectiveAttributes' is defined as a synonym for the OID 2.5.18.0. It would typically be used as a value of the collectiveExclusions attribute.

7. Other Optional Definitions

7.1. Attribute Types

7.1.1. Obsolete Attributes

Implementors must use modifyTimestamp and modifiersName instead.

```
( 0.9.2342.19200300.100.1.23 NAME 'lastModifiedTime' OBSOLETE
  SYNTAX 'UTCTime' )
```

```
( 0.9.2342.19200300.100.1.24 NAME 'lastModifiedBy' OBSOLETE
  EQUALITY distinguishedNameMatch SYNTAX 'DN' )
```

7.2. Syntaxes

7.2.1 MHSORAddress

Values of type MHSORAddress are encoded as strings, according to the format defined in [\[10\]](#).

7.2.2 DLSubmitPermission

Values of type DLSubmitPermission are encoded as strings, according to the following BNF:

```
<dlsubmit-perm> ::= <dlgroup_label> ':' <dlgroup-value>
                  | <dl-label> ':' <dl-value>
```

```
<dlgroup-label> ::= 'group_member'
```

```
<dlgroup-value> ::= <name>
```

```
<name> ::= an encoded Distinguished Name
```

```
<dl-label> ::= 'individual' | 'dl_member' | 'pattern'
```

```
<dl-value> ::= <orname>
```



```
<orname> ::= <address> '#' <dn>
           | <address>

<address> ::= <add-label> ':' <oraddress>

<dn> ::= <dn-label> ':' <name>

<add-label> = 'X400'

<dn-label> = 'X500'
```

where <oraddress> is as defined in [RFC 1327](#).

[7.3.](#) Object Classes

[7.3.1.](#) Obsolete Classes

```
( 0.9.2342.19200300.100.4.22 NAME 'oldQualityLabelledData' SUP top
  STRUCTURAL MUST dSAQuality
  MAY ( subtreeMaximumQuality $ subtreeMinimumQuality ) )
```

The oldQualityLabelledData object class is historical and must not be used for defining new objects.

[7.3.2.](#) extensibleObject

```
( 1.3.6.1.4.1.1466.101.120.111 NAME 'extensibleObject'
  SUP top AUXILIARY )
```

This class, if present in an entry, permits that entry to optionally hold any attribute. The MAY attribute list of this class is implicitly the set of all attributes known to the server. The mandatory attributes of the other object classes of this entry are still required to be present.

Note that not all servers will implement this object class, and those which do not will reject requests to add entries which contain this object class, or modify an entry to add this object class.

[7.4.](#) Matching Rules

[7.4.1.](#) caseIgnoreIA5SubstringsMatch

```
( 1.3.6.1.4.1.1466.109.114.3
  NAME 'caseIgnoreIA5SubstringsMatch' SYNTAX 'IA5String' )
```

This matching rule may be used to compare components of an IA5 string against an attribute whose values have IA5 string syntax.

[8.](#) Security Considerations

Security issues are not discussed in this memo.

Wahl, Coulbeck, Howes & Kille

[Page 37]

9. Acknowledgements

This document is based substantially on [RFC 1778](#), written by Tim Howes, Steve Kille, Wengyik Yeong and Colin Robbins.

Many of the attribute syntax encodings defined in this document are adapted from those used in the QUIPU and the IC R3 X.500 implementations. The contributions of the authors of both these implementations in the specification of syntaxes in this document are gratefully acknowledged.

10. Authors Addresses

Mark Wahl
Critical Angle Inc.
4815 West Braker Lane #502-385
Austin, TX 78759
USA

EMail: M.Wahl@critical-angle.com

Andy Coulbeck
ISODE Consortium
The Dome, The Square
Richmond TW9 1DT
United Kingdom

Phone: +44 181-332-9091
E-Mail: A.Coulbeck@isode.com

Tim Howes
Netscape Communications Corp.
501 E. Middlefield Rd
Mountain View, CA 94043
USA

Phone: +1 415 254-1900
E-Mail: howes@netscape.com

Steve Kille
ISODE Consortium
The Dome, The Square
Richmond
TW9 1DT
UK

Phone: +44-181-332-9091
EMail: S.Kille@isode.com

Wahl, Coulbeck, Howes & Kille

[Page 38]

11. Bibliography

- [1] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (Version 3)", INTERNET-DRAFT <[draft-ietf-asid-ldapv3-protocol-03.txt](#)>, October 1996.
- [2] The Directory: Selected Attribute Types. ITU-T Recommendation X.520, 1993.
- [3] The Directory: Models. ITU-T Recommendation X.501, 1993.
- [4] P. Barker, S. Kille, "The COSINE and Internet X.500 Schema", [RFC 1274](#), November 1991.
- [5] M. Wahl, S. Kille, "A UTF-8 String Representation of Distinguished Names", INTERNET-DRAFT <[draft-ietf-asid-ldapv3-dn-00.txt](#)>, August 1996.
- [6] S. Kille, "A String Representation for Presentation Addresses", [RFC 1278](#), University College London, November 1991.
- [7] Terminal Equipment and Protocols for Telematic Services - Standardization of Group 3 facsimile apparatus for document transmission. CCITT, Recommendation T.4.
- [8] JPEG File Interchange Format (Version 1.02). Eric Hamilton, C-Cube Microsystems, Milpitas, CA, September 1, 1992.
- [9] The Directory: Selected Object Classes. ITU-T Recommendation X.521, 1993.
- [10] H. Alvestrand, S. Kille, R. Miles, M. Rose, S. Thompson, "Mapping between X.400 and [RFC-822](#) Message Bodies", [RFC 1495](#), August 1993.
- [11] M. Davis, UTF-8, (WG2 N1036) DAM for ISO/IEC 10646-1.
- [12] Universal Multiple-Octet Coded Character Set (UCS) - Architecture and Basic Multilingual Plane, ISO/IEC 10646-1 : 1993.
- [13] The Directory: Authentication Framework. ITU-T Recommendation X.509 (1993).
- [14] Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation. ITU-T Recommendation X.680, 1994.

[<draft-ietf-asid-ldapv3-attributes-03.txt>](#)

Expires: April 1997

Wahl, Coulbeck, Howes & Kille

[Page 39]